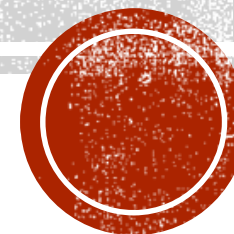


PYTHON

Tutora: Taís Bellini



A TUTORA

- Meu nome é Tais, tenho 22 anos e estou me formando em Ciência da Computação na UFRGS em Porto Alegre. Trabalho como Engenheira de Software na HP e uso o Python no trabalho 😊
- Apesar de a minha mãe ser da área de TI, nunca tive vontade de seguir nessa área até eu chegar no terceiro ano. Hoje em dia, não me imagino em outra profissão 😊.



A LINGUAGEM

- O Python é uma linguagem ótima para se começar a programar. É muito simples e intuitiva.
- É uma linguagem INTERPRETADA. Mas o que significa isso?
 - Temos linguagens COMPILADAS, que são TRADUZIDAS pelo COMPILADOR, para linguagem de máquina e então são executadas diretamente pelo processador ou sistema operacional.
Ex: C, C++.
 - E temos as linguagens INTERPRETADAS, que são executadas por um programa chamado “interpretador”, e então são executadas pelo processador ou sistema operacional.
Ex: C#, JavaScript, e, claro, Python.



O TUTORIAL

- Então, o que vamos aprender nesse tutorial?
 - Instalação
 - Como Python é uma linguagem interpretada, vamos instalar o interpretador dela no nosso ambiente. Vou mostrar como se faz em Windows, Linux e Mac OSX.
 - Sintaxe básica do Python
 - Vamos passar rapidamente pelo básico da linguagem, como declaração de variáveis, classes, métodos, etc.
 - Exemplos de fixação da linguagem.
 - Exercício Final
 - Vamos criar “Jogo da Forca” para exercitar a linguagem e terminar o tutorial com um projeto completo.



INSTALAÇÃO WINDOWS

- Vá em: <https://www.python.org/downloads/> e baixe a última versão que começa com 2 (2.x.x).
 - Por que 2? Essa versão, apesar de menos recente, ainda é mais completa!
- Execute o instalador python-2.7.11.msi e apenas vá clicando “Next”.
 - **DETALHE:** Quando chegar nessa janela:
 - Lembre de marcar a última opção!
Assim, vai facilitar para que você use o python pelo terminal 😊



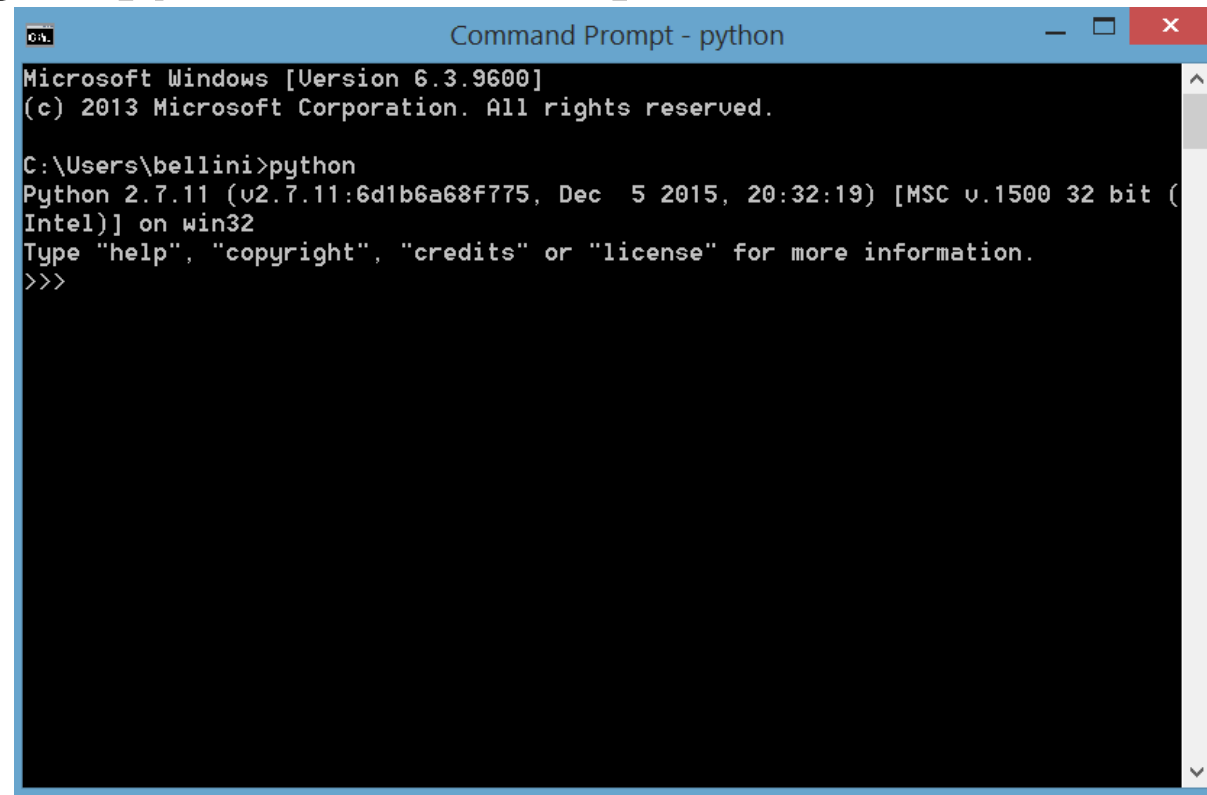
INSTALAÇÃO MAC E LINUX

- Como não tenho acesso a máquinas Mac e Linux, não tenho como colocar prints, mas aqui tem um tutorial com as informações necessárias:
- <http://wiki.python.org.br/InicieSe>
- Qualquer problema pode me contatar que trabalhamos juntas para resolver ☺



INSTALAÇÃO

- Para ter certeza de que está instalado, abra o programa chamado “Terminal” (ou “cmd”) e digite “python”. Isso deve aparecer:



```
Command Prompt - python
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\bellini>python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

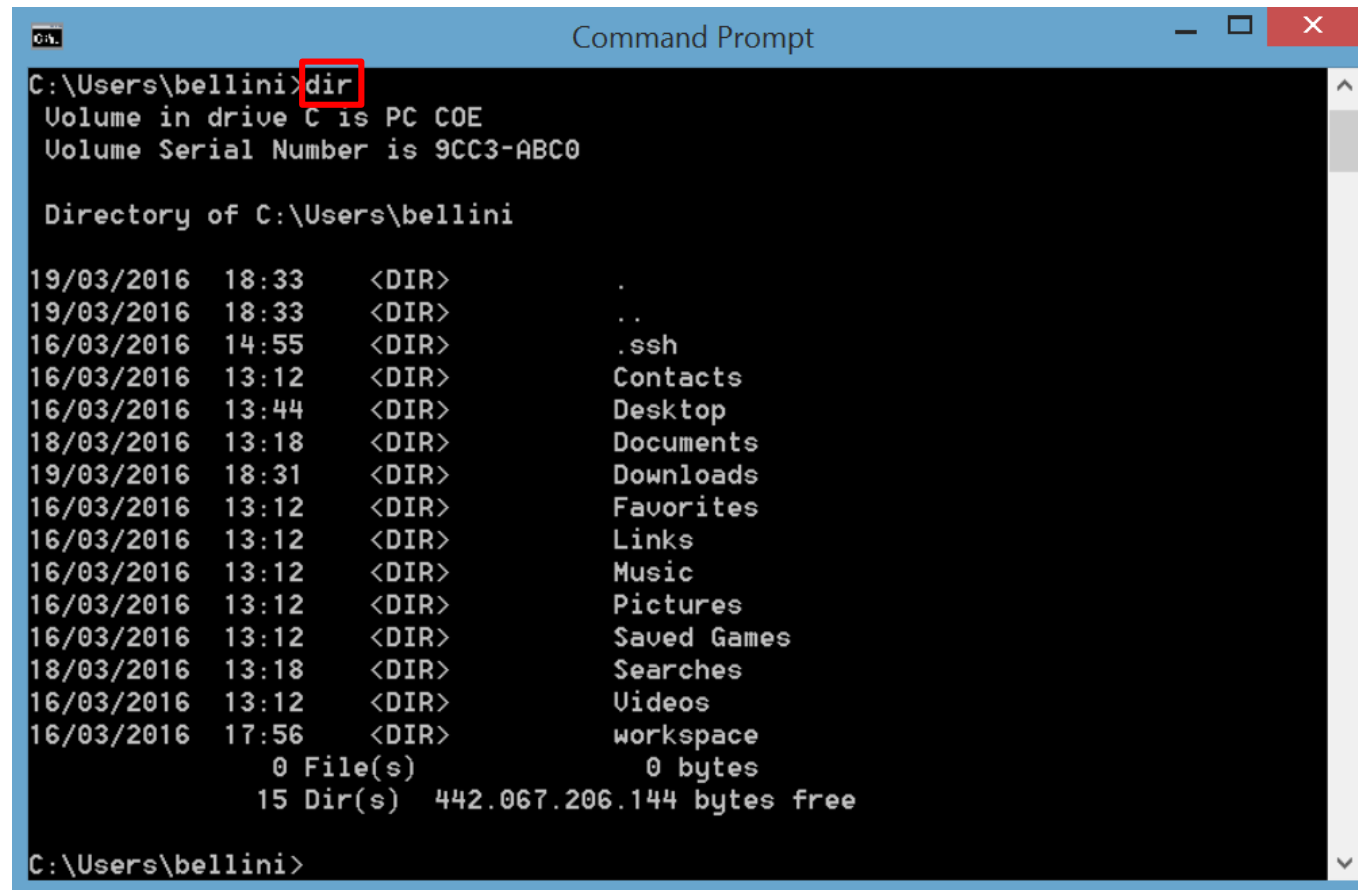


TERMINAL

- Isso que aparece na tela do Terminal é o **interpretador**. Podemos escrever código diretamente por ali e o nosso programa será executado. Use e abuse dele para praticar.
- O python é muito utilizado através de linha de comando no Terminal, mas você não precisa ser uma “expert” nisso para usar o Python.
- A única coisa que precisa saber é como chegar na Pasta que está o seu arquivo. Para isso, você vai usar o comando “cd”.
- O Terminal geralmente abre na pasta do usuário (no meu caso: “bellini”).
- Para listar as pastas e arquivos do diretório que você está, digite “dir” no Windows ou “ls” no Mac ou Linux.



TERMINAL



```
Command Prompt
C:\Users\bellini>dir
Volume in drive C is PC COE
Volume Serial Number is 9CC3-ABC0

Directory of C:\Users\bellini

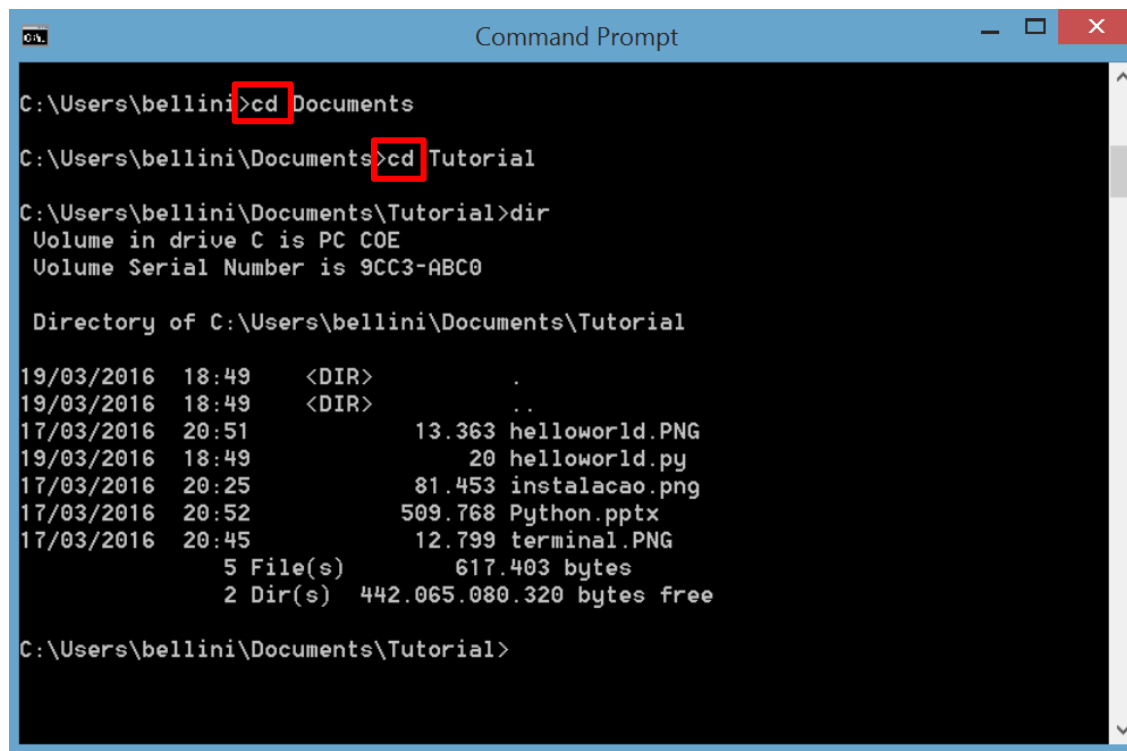
19/03/2016  18:33    <DIR>          .
19/03/2016  18:33    <DIR>          ..
16/03/2016  14:55    <DIR>          .ssh
16/03/2016  13:12    <DIR>          Contacts
16/03/2016  13:44    <DIR>          Desktop
18/03/2016  13:18    <DIR>          Documents
19/03/2016  18:31    <DIR>          Downloads
16/03/2016  13:12    <DIR>          Favorites
16/03/2016  13:12    <DIR>          Links
16/03/2016  13:12    <DIR>          Music
16/03/2016  13:12    <DIR>          Pictures
16/03/2016  13:12    <DIR>          Saved Games
18/03/2016  13:18    <DIR>          Searches
16/03/2016  13:12    <DIR>          Videos
16/03/2016  17:56    <DIR>          workspace
               0 File(s)                0 bytes
               15 Dir(s)  442.067.206.144 bytes free

C:\Users\bellini>
```



TERMINAL

- Os meus arquivos do tutorial estão da Pasta “Documents/Tutorial”, então, vou usar o comando “cd” para navegar até lá:



```
C:\Users\bellini>cd Documents
C:\Users\bellini\Documents>cd Tutorial
C:\Users\bellini\Documents\Tutorial>dir
Volume in drive C is PC C0E
Volume Serial Number is 9CC3-ABC0

Directory of C:\Users\bellini\Documents\Tutorial

19/03/2016  18:49    <DIR>          .
19/03/2016  18:49    <DIR>          ..
17/03/2016  20:51             13.363 helloworld.PNG
19/03/2016  18:49              20 helloworld.py
17/03/2016  20:25             81.453 instalacao.png
17/03/2016  20:52            509.768 Python.pptx
17/03/2016  20:45             12.799 terminal.PNG
               5 File(s)              617.403 bytes
               2 Dir(s)  442.065.080.320 bytes free

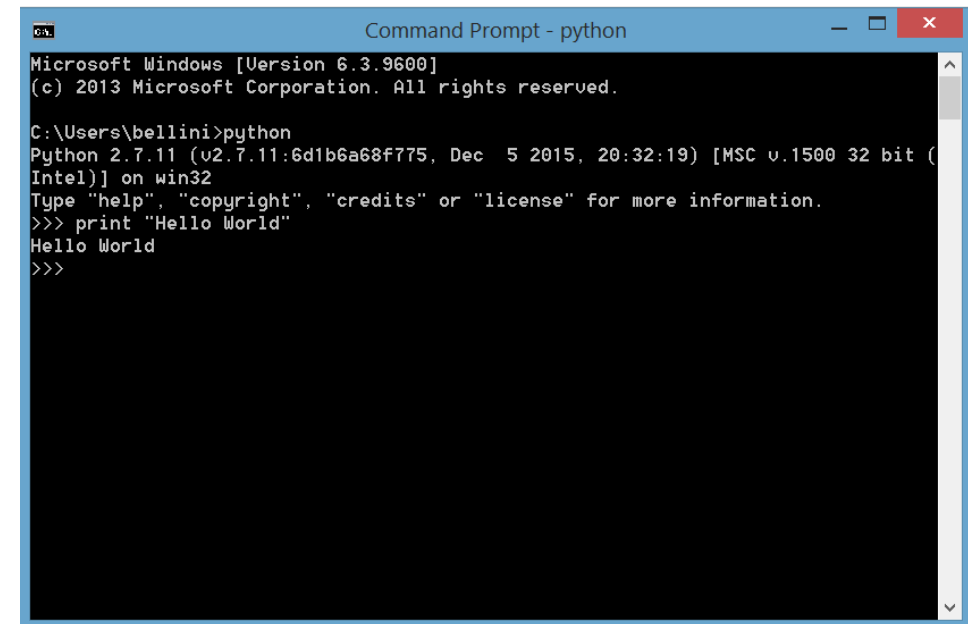
C:\Users\bellini\Documents\Tutorial>
```



HELLO WORLD!

- Todo bom tutorial tem que ter o famoso “Hello World” para introduzir a linguagem e mostrar como criar o seu primeiro programa.
- O Python é TÃO simples, que chega a ser sem graça o nosso “Hello World” hehe, mas vamos lá:
- Abra **interpretador** e digite:

```
print “Hello World!”
```



```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\bellini>python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print "Hello World"
Hello World
>>>
```



PROGRAMANDO EM PYTHON

- Podemos usar o Python de duas formas: diretamente no interpretador (por linha de comando no Terminal), como acabamos de fazer, ou escrevendo o programa em um editor de texto e depois executando no terminal.
- Como Editor de Texto, eu gosto de usar o Sublime Text (<https://www.sublimetext.com>) e é nele que vou mostrar os exemplos, mas existem vários outros que você pode usar 😊
- Para começar um programa em Python, crie um arquivo com extensão “.py” no Sublime e escreva o mesmo comando “print ‘Hello World!’”.
- Abra o Terminal e navegue até a Pasta que se encontra o seu arquivo.
- Execute: `python <nome_do_arquivo>.py`
- O meu arquivo se chama “helloworld.py”, portanto eu digitei “python helloworld.py”



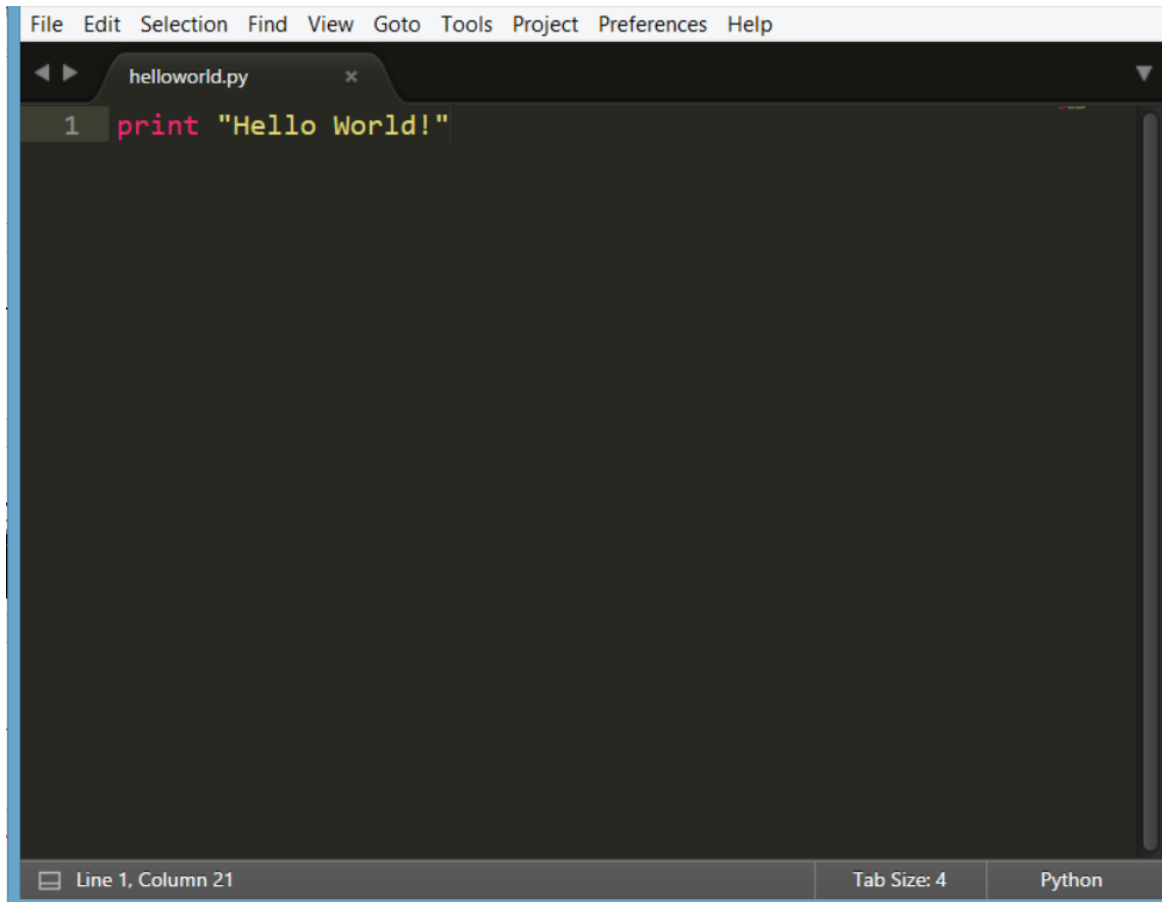
COMENTÁRIOS

- Comentários são partes do código que servem só para o uso dos programadores e não são processados pelo computador.
- Podemos ter comentários de uma linha usando o '#', ou comentários com mais linhas usando três aspas simples:

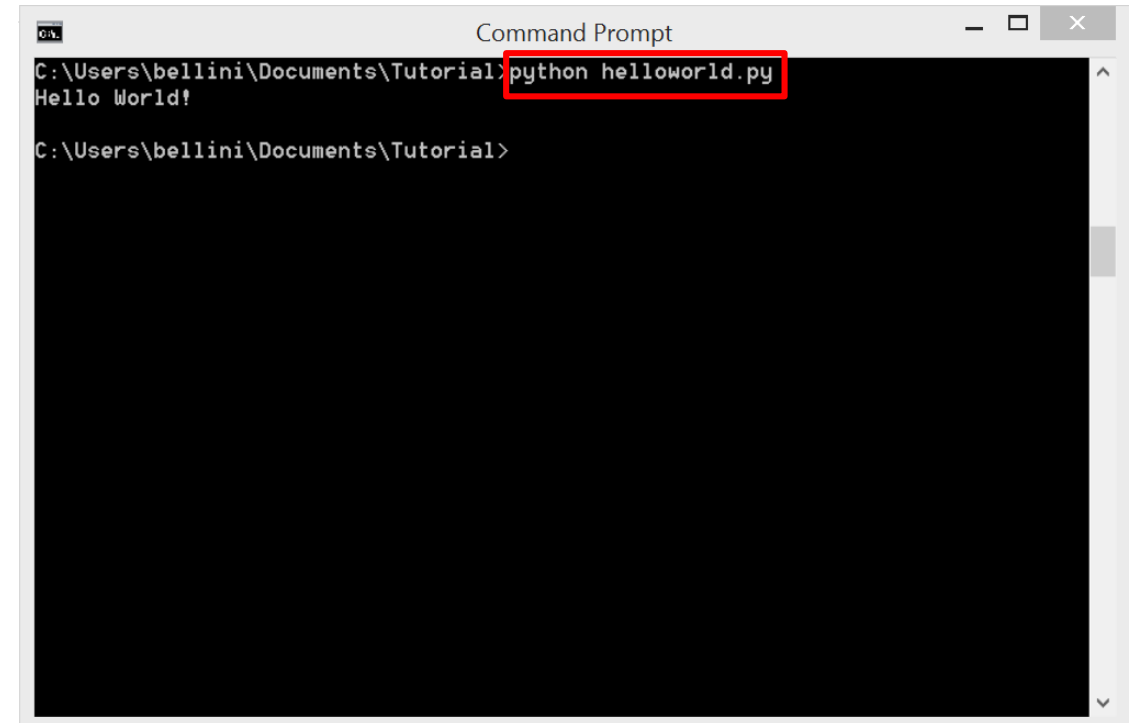
```
1  #isso eh u comentario de uma linha
2  '''
3      Isso eh um comentario
4      de varias
5      linhas
6  '''
```



PROGRAMANDO EM PYTHON



A screenshot of a Python IDE window. The title bar shows 'File Edit Selection Find View Goto Tools Project Preferences Help'. The editor has a tab for 'helloworld.py'. The code inside is a single line: `1 print "Hello World!"`. The status bar at the bottom indicates 'Line 1, Column 21', 'Tab Size: 4', and 'Python'.



A screenshot of a Windows Command Prompt window. The title bar says 'Command Prompt'. The command prompt shows the directory `C:\Users\bellini\Documents\Tutorial`. The command `python helloworld.py` has been entered and executed, resulting in the output `Hello World!`. The command `python helloworld.py` is highlighted with a red rectangle.



VARIÁVEIS

- Para manipular dados, usamos variáveis.
- O ideal é que se dê nomes significativos para elas, para não nos perdermos em programas muito grandes.
- Muitas linguagens exigem que se defina explicitamente o **tipo** da variável: int, double, string, etc., mas o Python já entende o tipo dela automaticamente.
- Quando o valor da variável é um **caractere** ou uma **string** (sequência de caracteres), utilizamos aspas (simples ou duplas).
- Temos também o booleano, que é **True** ou **False**.
- Quando é um **número**, colocamos apenas ele.
- Ex:

```
nome = "Tais"  
idade = 22  
sexo = "F"  
is_Feliz = True
```



ENTRADA E SAÍDA

- Uma funcionalidade muito útil e usada em linguagens de programação é entrada e saída de dados.
- Entrada é quando nós fornecemos dados para o programa.
- Saída é quando o programa nos mostra dados na tela.
- Como fazer a saída nós já aprendemos, é o `print` !!
- Para pegar a entrada, usamos o `input()` e colocamos em uma variável:

```
numero = input()
```

- Podemos ainda colocar algum texto com instruções:

```
numero = input("Digite um numero: ")
```

- Para strings, usamos o `raw_input()`:

```
nome = raw_input("Digite seu nome:")
```



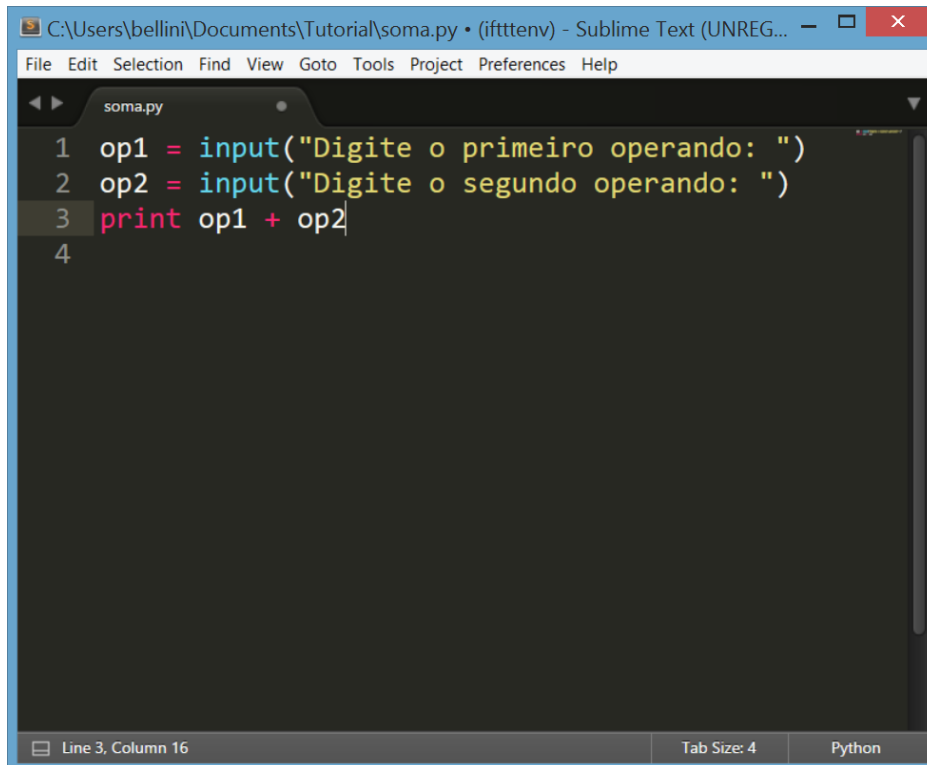
OPERAÇÕES ARITMÉTICAS

- O Python suporta todas as operações básicas:
- Adição: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Resto da Divisão: %
- Potência: **

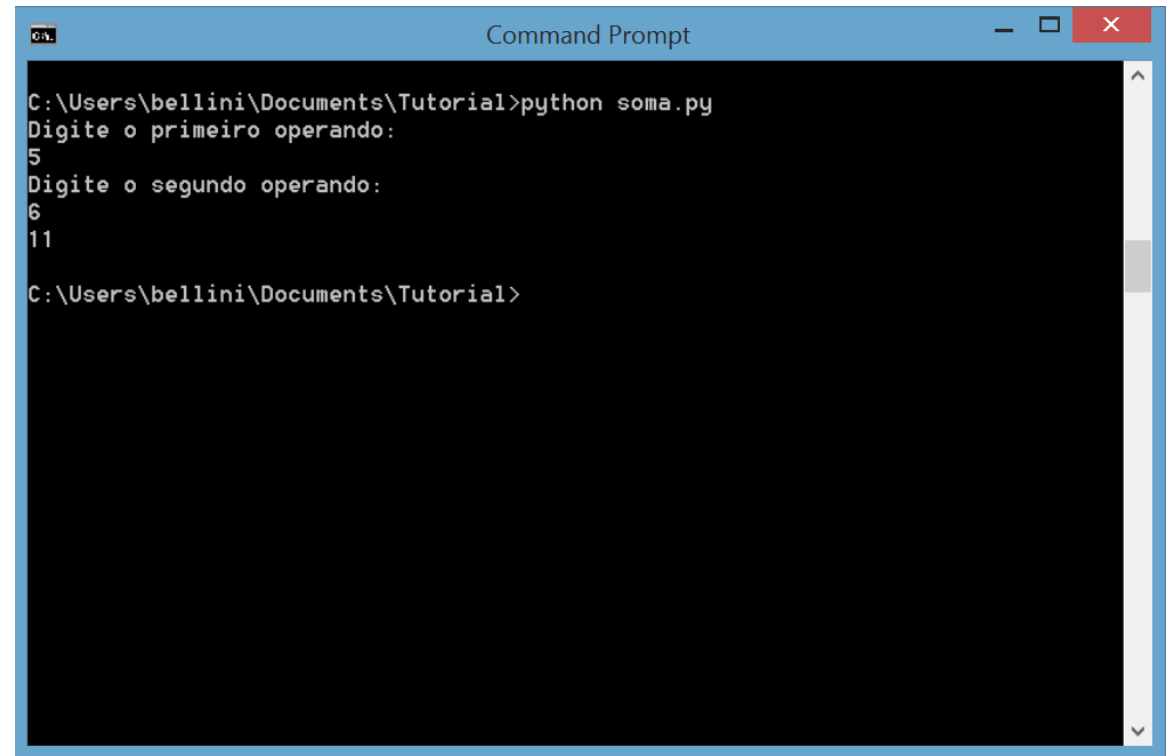


EXEMPLO

- Para exemplificar esses conceitos, vamos fazer um pequeno programinha que recebe do usuário dois operandos, e imprime na tela a SOMA deles.



```
C:\Users\bellini\Documents\Tutorial\soma.py • (iffttenv) - Sublime Text (UNREG...  
File Edit Selection Find View Goto Tools Project Preferences Help  
soma.py  
1 op1 = input("Digite o primeiro operando: ")  
2 op2 = input("Digite o segundo operando: ")  
3 print op1 + op2  
4  
Line 3, Column 16 Tab Size: 4 Python
```



```
Command Prompt  
C:\Users\bellini\Documents\Tutorial>python soma.py  
Digite o primeiro operando:  
5  
Digite o segundo operando:  
6  
11  
C:\Users\bellini\Documents\Tutorial>
```



OPERAÇÕES DE COMPARAÇÃO

- No Python, podemos comparar duas variáveis e receber um resultado de **verdadeiro** ou **falso** para a comparação.
- $a == b$: retorna verdadeiro se **a** for igual a **b**
- $a != b$: retorna verdadeiro se **a** for diferente de **b**
- $a > b$: retorna verdadeiro se **a** for maior que **b**
- $a >= b$: retorna verdadeiro se **a** for maior ou igual a **b**
- $a < b$: retorna verdadeiro se **a** for menor que **b**
- $a <= b$: retorna verdadeiro se **a** for maior ou igual a **b**



OPERAÇÕES LÓGICAS

- As operações lógicas básicas também são suportadas no Python, de forma muito intuitiva:
- **and** : operação “e” lógica
- **or** : operação “ou” lógica
- **not** : operação de negação



CONDICIONAIS

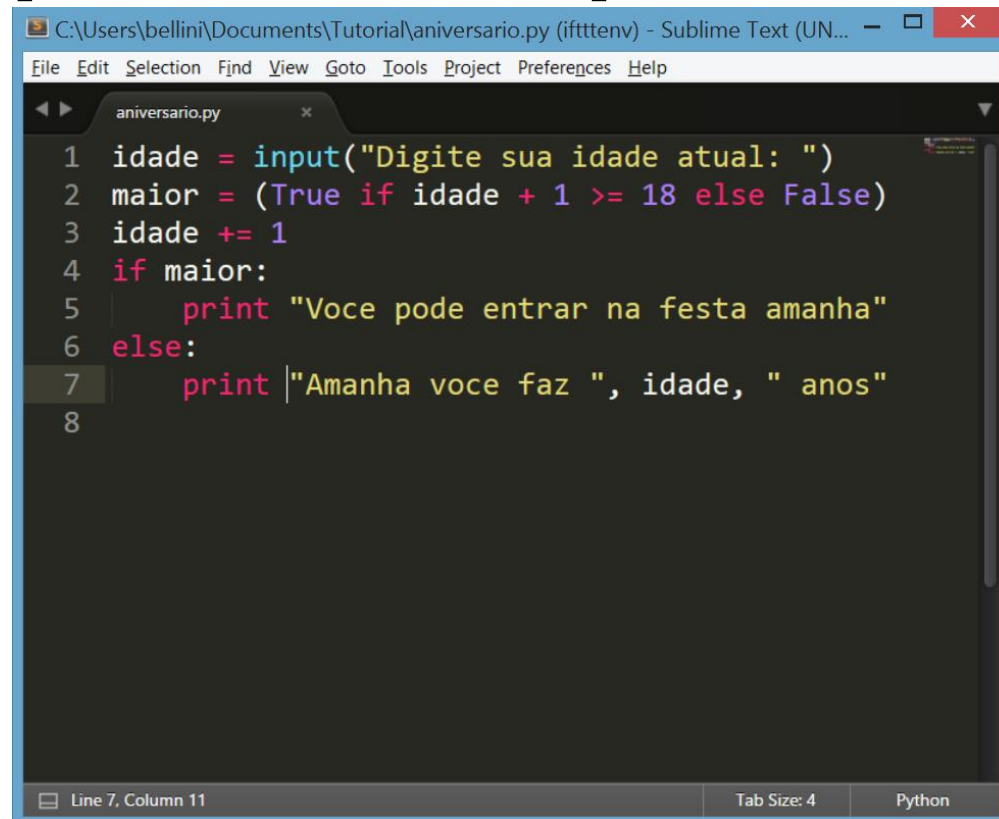
- Para fazer decisões, utilizamos as operações condicionais “if...else” (se... então)
- Podemos usar operações aninhadas também: if...else if else
- Temos ainda o operador ternário, que, no Python, é como se fosse uma frase: “Faça isso se essa condição for verdadeira, se não faça aquilo”.
 - Ex: Dizer que a pessoa é adulta se a idade for maior que 18, senão dizer que é menor:

“adulto” **if** idade > 18 **else** “menor”

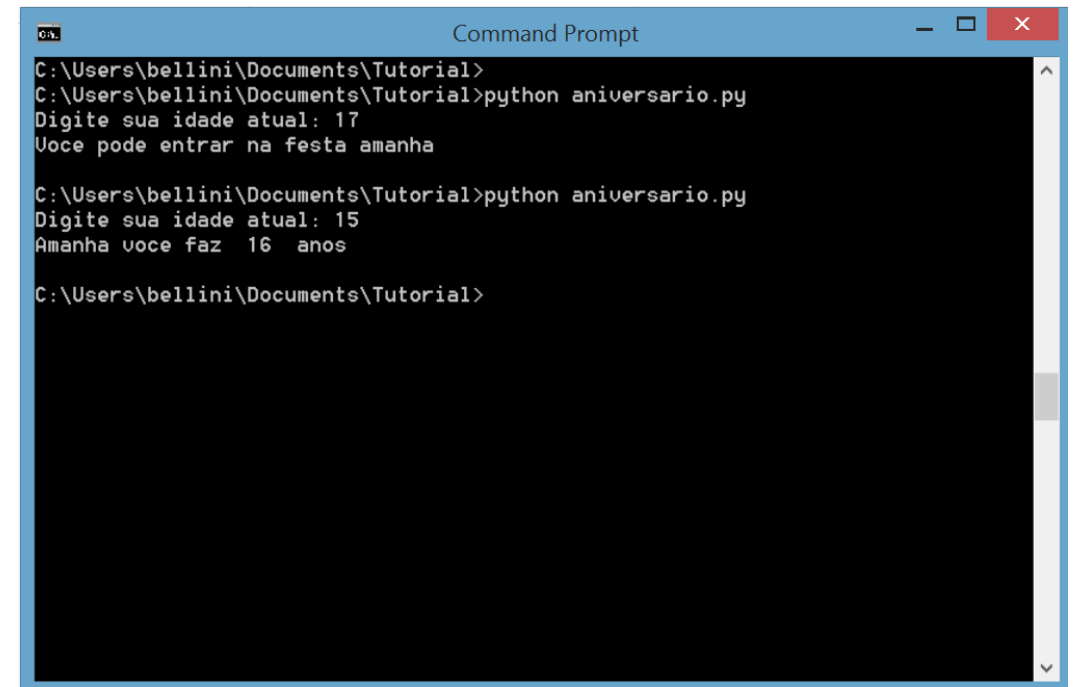


EXEMPLO

- Um programa que você executa um dia antes do seu aniversário para saber se poderá entrar na festa que tem no dia seguinte:



```
C:\Users\bellini\Documents\Tutorial\aniversario.py (iftttenv) - Sublime Text (UN...  
File Edit Selection Find View Goto Tools Project Preferences Help  
aniversario.py  
1 idade = input("Digite sua idade atual: ")  
2 maior = (True if idade + 1 >= 18 else False)  
3 idade += 1  
4 if maior:  
5     print "Voce pode entrar na festa amanha"  
6 else:  
7     print "Amanha voce faz ", idade, " anos"  
8  
Line 7, Column 11 Tab Size: 4 Python
```



```
Command Prompt  
C:\Users\bellini\Documents\Tutorial>  
C:\Users\bellini\Documents\Tutorial>python aniversario.py  
Digite sua idade atual: 17  
Voce pode entrar na festa amanha  
  
C:\Users\bellini\Documents\Tutorial>python aniversario.py  
Digite sua idade atual: 15  
Amanha voce faz 16 anos  
  
C:\Users\bellini\Documents\Tutorial>
```



OBSERVAÇÕES

▪ IDENTIFICAÇÃO

- Identação é o recuo do texto em relação à margem, como o TAB, por exemplo.
- É **MUITO** importante no Python.
- Blocos de comandos são separados por IDENTIFICAÇÃO. Observe que no **if** do exemplo, a ação está indentada abaixo dele.

▪ OPERAÇÃO +=:

- Observe que ao invés de eu somar a idade com `idade = idade + 1`, eu apenas usei `+=`.
- Isso pode ser feito com qualquer operador aritmético.

▪ PRINT DE VARIÁVEIS

- Observe que eu posso alternar entre imprimir na tela strings ou variáveis usando a vírgula.



LOOPS

- O python possui os loops padrão “for” e “while”.
- Além disso, tem a função `range()` , que faz uma progressão aritmética.
- Exemplos no **interpretador**:

```
Command Prompt - python
C:\Users\bellini\Documents\Tutorial>python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> for i in range(10):
...     print "Numero ", i
...
Numero 0
Numero 1
Numero 2
Numero 3
Numero 4
Numero 5
Numero 6
Numero 7
Numero 8
Numero 9
>>>
```

```
Command Prompt - python
C:\Users\bellini\Documents\Tutorial>
C:\Users\bellini\Documents\Tutorial>python
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> i = 10
>>> while (i > 0):
...     print "Numero ", i
...     i -= 1
...
Numero 10
Numero 9
Numero 8
Numero 7
Numero 6
Numero 5
Numero 4
Numero 3
Numero 2
Numero 1
>>>
```



ESTRUTURAS DE DADOS

■ Lista

■ Declaração

- `lista = []`

■ Operações:

- `lista.append("item 1")` – adiciona um item na lista
- `lista.remove("item 1")` – remove um item da lista
- `lista[i]` – pega o elemento na posição “i” da lista

■ Veja, na imagem, que podemos iterar pela lista usando o **for**

- Veja, também, que é muito fácil identificar se um elemento está ou não na lista

```
>>> lista = []
>>> lista.append("item1")
>>> lista.append("item2")
>>> for item in lista:
...     print item
...
item1
item2
>>> lista.remove("item2")
>>> for item in lista:
...     print item
...
item1
>>> if "item1" in lista:
...     print "Tem item1!"
...
Tem item1!
>>> if "item2" not in lista:
...     print "Nao tem item2!"
...
Nao tem item2!
```



ESTRUTURAS DE DADOS

- Dicionários

- Declaração

- dict = {}

- Operações:

- dict["item1"] = "banana" – adiciona um item que possui chave "item1" e valor "banana"

- del dict["item1"] = remove o item 1

- Podemos ver no exemplo que é possível colocar listas no dicionário

- Podemos usar comandos como **for** e **if** no dicionário também

```
>>> comidas = {}
>>> comidas["banana"] = "fruta"
>>> comidas["arroz"] = "graos"
>>> for comida in comidas:
...     print comida, " eh do tipo ", comidas[comida]
...
arroz eh do tipo  graos
banana eh do tipo  fruta
>>> ingredientes = {}
>>> ingredientes["sanduiche"] = ["pao", "queijo", "alface", "tomate"]
>>> ingredientes["panqueca"] = ["farinha", "manteiga", "ovo", "canela"]
>>> print ingredientes
{'panqueca': ['farinha', 'manteiga', 'ovo', 'canela'], 'sanduiche': ['pao', 'queijo', 'alface', 'tomate']}
```



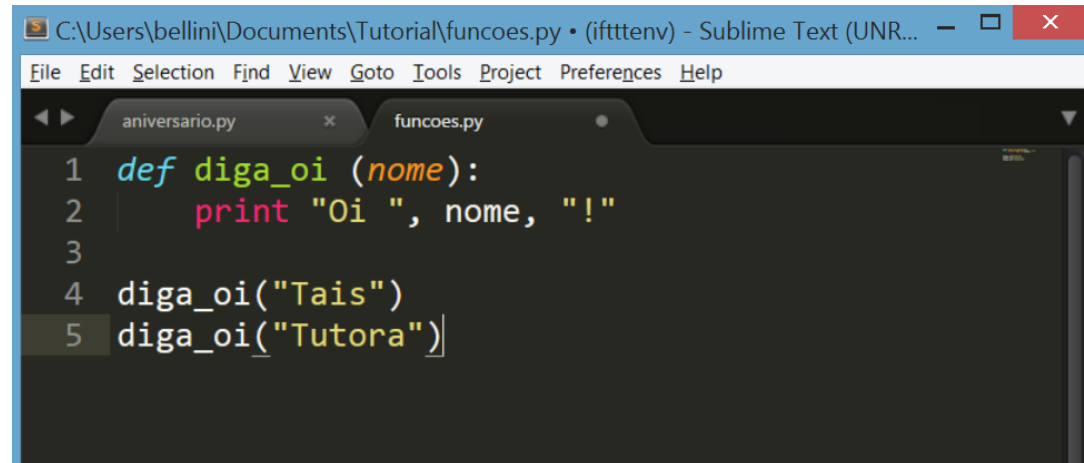
FUNÇÕES (MÓDULOS)

- Uma **função** é um trecho de código que faz uma ação específica e pode, ou não, retornar um resultado.
- Podemos também passar valores como parâmetros para que a função possa usá-los.
- Seu objetivo é evitar que trechos de código sejam muito repetidos.
- Por exemplo: se queremos fazer dez somas diferentes durante um mesmo programa, fazemos uma função que recebe dois operandos e retorna a soma deles. Assim, o código que faz a soma é escrito uma vez só, ao invés de 10.

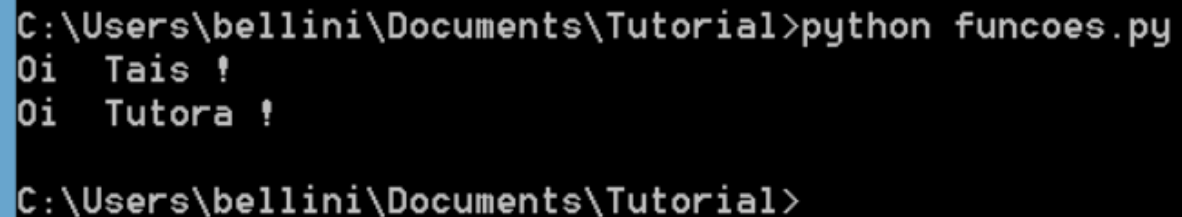


FUNÇÕES (MÓDULOS)

- Para declarar uma função, basta usar a palavra “def” e definir um nome para ela:
 - Observe a importância da indentação
 - Observe que posso passar dados de qualquer **tipo** como argumento da função (neste caso, é uma string chamada “nome”).
 - Para chamar a função, basta escrever o nome dela e passar o(s) parâmetros entre parênteses.



```
C:\Users\bellini\Documents\Tutorial\funcoes.py • (iftttenv) - Sublime Text (UNR...  
File Edit Selection Find View Goto Tools Project Preferences Help  
aniversario.py x funcoes.py  
1 def diga_oi (nome):  
2     print "Oi ", nome, "!"  
3  
4 diga_oi("Tais")  
5 diga_oi("Tutora")
```



```
C:\Users\bellini\Documents\Tutorial>python funcoes.py  
Oi Tais !  
Oi Tutora !  
  
C:\Users\bellini\Documents\Tutorial>
```



UTILIDADES

- O Python tem algumas funções pré-prontas (como o `range()`) que facilitam a nossa vida:
 - `lower()` – passa a letra para minúsculo
 - `len(s)` – informa o tamanho da string “s”
 - `max(a, b, ...)` – informa qual é o maior valor
 - `min(a, b, ...)` – informa qual é o menor valor
 - `split()` – separa uma string

```
Python 2.7.11 (v2.7.11:6d1b6a68f775, Dec 5 2015, 20:32:19) [MSC v.1500 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> "TUTORA".lower()
'tutora'
>>> len("tutora")
6
>>> max(2, 4, 12)
12
>>> min(2, 4, 12)
2
>>> "eu amo python".split()
['eu', 'amo', 'python']
```



UTILIDADES

- Strings:
 - Podemos usar a seguinte sintaxe para cortar strings: [:]
 - Tenha em mente que uma **string** é uma LISTA de caracteres
 - s = "tutora"
 - s[:3] – retorna a string do começo até a posição 2
 - s[3:] – retorna a string a partir da posição 3 até o final
 - s[2:4] - retorna a string da posição 2 até a posição 3
 - * lembrando que as posições são 0, 1, 2,
 - Podemos usar o + para concatenar strings

```
>>> s = "tutora"
>>> s[:3]
'tut'
>>> s[3:]
'ora'
>>> s[2:4]
'to'
>>> "tut" + "ora"
'tutora'
```



MÃO NA MASSA

- Vamos agora aplicar os conhecimentos obtidos até agora e construir um “Jogo da Forca” usando o Python.
- Para facilitar a nossa vida, não teremos o desenho do boneco sendo enforcado, apenas os espaços em branco para as letras da palavra e a lista de letras já usadas que estão erradas:

```
F O R C A  
Letras erradas:  
_ _ _ _ _  
Chute uma letra.
```

- Vamos começar fazendo um print do nome do jogo:

```
print " F O R C A "
```



MÃO NA MASSA

- Agora vamos definir estruturas e variáveis que precisamos:
 - Uma lista de letras usadas, separando entre “certas” e “erradas”;
 - Serão duas listas
 - Um booleano definindo se já acabou o jogo;
 - Um inteiro com o número de chances que o jogador vai ter.

```
print " F O R C A "  
  
erradas = []  
certas = []  
fim = False  
chances = 7
```



MÃO NA MASSA

- Precisamos também de uma palavra secreta para ser adivinhada:
 - Vamos fazer uma lista de palavras e no começo de cada jogo escolheremos uma aleatória.
 - Usamos o `split()` para criar a lista a partir de uma string.
 - Usamos a função **random** nativa do Python para escolher a palavra. Ela será escolhida numa função separada, declarada no começo do programa (boas práticas).
 - Para usar o **random**, é preciso importá-lo.
 - Veja (na próxima página) que escolhemos um número aleatório entre 0 e o tamanho da nossa lista de palavras **menos** 1. Isso porque o índice da lista começa em zero. Portanto, se temos 5 palavras, teremos índices: 0, 1, 2, 3, 4.



MÃO NA MASSA

```
1 import random
2
3 def getPalavra(ListaDePalavras):
4     # Retorna uma palavra aleatoria da lista
5     index = random.randint(0, len(ListaDePalavras) - 1)
6     return ListaDePalavras[index]
7
8 print " F O R C A "
9
10 erradas = []
11 certas = []
12 fim = False
13 chances = 7
14
15 palavras = 'batata barriga bocejo camelo cachorro candelabro cobra cadeira sapato paralelepipedo macaco'.split()
16 palavra = getPalavra(palavras) |
```



MÃO NA MASSA

- Agora, vamos começar o jogo.
 - Pense que o nosso jogo fica acontecendo enquanto a pessoa tem jogadas a fazer. Portanto, todo o jogo ficará dentro de um **while**.

```
1 import random
2
3 def getPalavra(listaDePalavras):
4     # Retorna uma palavra aleatoria da lista
5     index = random.randint(0, len(listaDePalavras) - 1)
6     return listaDePalavras[index]
7
8 print " F O R C A "
9
10 erradas = []
11 certas = []
12 fim = False
13 chances = 7
14
15 palavras = 'batata barriga bocejo camelo cachorro candelabro cobra cadeira sapato paralelepipedo macaco'.split()
16 palavra = getPalavra(palavras)
17
18 while fim == False:
19
```



MÃO NA MASSA

- Precisamos de uma função que mostre o status do jogo
 - Para isso, precisamos também de uma lista de espaços em branco do tamanho da palavra para mostrar.
 - Teremos a função **mostra** que:
 - Mostrará as letras erradas já chutadas
 - Receberemos como argumento a lista de letras erradas
 - Usaremos um **for in** para mostrar

```
def mostra(erradas, certas, palavra):  
    print 'Letras erradas:',  
    for letra in erradas:  
        print letra,  
    print " "
```

- Observe que a vírgula no final do print faz com que tudo seja impresso na mesma linha



MÃO NA MASSA

- Mostrará os espaços em branco contendo as letras certas
 - Definiremos uma lista de “_”
 - Usaremos a lista de letras certas e a palavra a ser adivinhada que foram passadas como argumento
 - Usaremos o corte de strings para colocar as letras no lugar certo:
 - Percorreremos a palavra a ser adivinhada letra a letra e, se a letra em questão está na lista de “certas”, pegamos os espaços em branco até ali, adicionamos a letra, e completamos com os espaços em branco faltantes.

```
espacos = '_' * len(palavra)

for i in range(len(palavra)): # substitui espacos com as letras corretas
    if palavra[i] in certas:
        espacos = espacos[:i] + palavra[i] + espacos[i+1:]

for letra in espacos: # mostra os espacos com as letras corretas
    print letra,
print " "|
```



MÃO NA MASSA

- Explicando o código:
 - Por exemplo: Se a palavra é **bola** e você já chutou a letra “o”, temos:
 - `espacos = _ _ _ _`
 - `Palavra[0] = “b”, palavra[1] = “o”, palavra[2] = “l”, palavra[3] = “a”`
 - A primeira iteração ($i = 0$) é a letra “b”, não acontece nada.
 - A segunda ($i = 1$) é a letra “o”, então: `espacos[:1] = _`, `palavra[1] = “o”`, `espacos[1+1:] = _ _`
 - O resultado é `_ o _ _`
 - Nas outras iterações não acontece nada



MÃO NA MASSA

- Função **mostra**

```
def mostra(erradas, certas, palavra):  
    print 'Letras erradas:',  
    for letra in erradas:  
        print letra,  
    print " "  
  
    espacos = '_' * len(palavra)  
  
    for i in range(len(palavra)): # substitui espacos com as letras corretas  
        if palavra[i] in certas:  
            espacos = espacos[:i] + palavra[i] + espacos[i+1:]  
  
    for letra in espacos: # mostra os espacos com as letras corretas  
        print letra,  
    print " "
```



MÃO NA MASSA

- Resultado até agora (veja o número das linhas para não se perder):

```
1 import random
2
3 def getPalavra(listaDePalavras):
4     # Retorna uma palavra aleatoria da lista
5     index = random.randint(0, len(listaDePalavras) - 1)
6     return listaDePalavras[index]
7
8 def mostra(erradas, certas, palavra):
9     print 'Letras erradas:',
10    for letra in erradas:
11        print letra,
12    print " "
13
14    espacos = '_' * len(palavra)
15
16    for i in range(len(palavra)): # substitui espacos com as letras corretas
17        if palavra[i] in certas:
18            espacos = espacos[:i] + palavra[i] + espacos[i+1:]
19
20    for letra in espacos: # mostra os espacos com as letras corretas
21        print letra,
22    print " "|
```



MÃO NA MASSA

```
40 print " F O R C A "
41
42 erradas = []
43 certas = []
44 fim = False
45 chances = 7
46
47 palavras = 'batata barriga bocejo camelo cachorro candelabro cobra cadeira sapato paralelepipedo macaco'.split()
48 palavra = getPalavra(palavras)
49
50 while fim == False:
51     mostra(erradas, certas, palavra)
52
```



MÃO NA MASSA

- Agora que já estamos mostrando a situação do jogo, temos que deixar o jogador chutar uma letra.
 - Faremos a função `getChute(usadas)` que recebe ambas as listas (**certas** e **erradas**) como argumento e faz a validação da entrada do usuário:

```
def getChute(usadas):  
    # Retorna a letra digitada e faz a validacao se eh uma letra valida  
    while True:  
        print('Chute uma letra.')  
        chute = input()  
        chute = chute.lower()  
        if len(chute) != 1:  
            print('Por favor informe uma unica palavra')  
        elif chute in usadas:  
            print('Voce ja usou essa letra. Tente novamente')  
        elif chute not in 'abcdefghijklmnopqrstuvwxyz':  
            print('Por favor use apenas LETRAS.')  
        else:  
            return chute
```



MÃO NA MASSA

- Observe o uso do **elif** que é uma abreviação de **else if**
- Aqui verificamos 3 coisas:
 - Se foi apenas uma letra;
 - Se a letra já foi usada;
 - Se É uma letra.

```
def getChute(usadas):  
    # Retorna a letra digitada e faz a validacao se eh uma letra valida  
    while True:  
        print('Chute uma letra.')  
        chute = input()  
        chute = chute.lower()  
        if len(chute) != 1:  
            print('Por favor informe uma unica palavra')  
        elif chute in usadas:  
            print('Voce ja usou essa letra. Tente novamente')  
        elif chute not in 'abcdefghijklmnopqrstuvwxyz':  
            print('Por favor use apenas LETRAS.')  
        else:  
            return chute
```



MÃO NA MASSA

- E então chamamos a função no **while** principal:

```
while fim == False:
    mostra(erradas, certas, palavra)

    # Pega a letra escolhida pelo jogador
    chute = getChute(erradas + certas)
```

- Por enquanto, se executarmos o nosso programa, obtemos o seguinte:

```
C:\Users\bellini\Documents\Tutorial>python forcaparts.py
F O R C A
Letras erradas:
_ _ _ _ _
Chute uma letra.
```



MÃO NA MASSA

- O código está assim (observe a o número das linhas para não se perder):

```
1  import random
2
3  def getPalavra(ListaDePalavras):
4      # Retorna uma palavra aleatoria da lista
5      index = random.randint(0, len(listaDePalavras) - 1)
6      return listaDePalavras[index]
7
8  def mostra(erradas, certas, palavra):
9      print 'Letras erradas:',
10     for letra in erradas:
11         print letra,
12     print " "
13
14     espacos = '_' * len(palavra)
15
16     for i in range(len(palavra)): # substitui espacos com as letras corretas
17         if palavra[i] in certas:
18             espacos = espacos[:i] + palavra[i] + espacos[i+1:]
19
20     for letra in espacos: # mostra os espacos com as letras corretas
21         print letra,
22     print " "
```



MÃO NA MASSA

```
23
24 def getChute(usadas):
25     # Retorna a letra digitada e faz a validacao se eh uma letra valida
26     while True:
27         print('Chute uma letra.')
28         chute = input()
29         chute = chute.lower()
30         if len(chute) != 1:
31             print('Por favor informe uma unica palavra')
32         elif chute in usadas:
33             print('Voce ja usou essa letra. Tente novamente')
34         elif chute not in 'abcdefghijklmnopqrstuvwxyz':
35             print('Por favor use apenas LETRAS.')
36         else:
37             return chute
38
39
```



MÃO NA MASSA

```
40 print " F O R C A "
41
42 erradas = []
43 certas = []
44 fim = False
45 chances = 7
46
47 palavras = 'batata barriga bocejo camelo cachorro candelabro cobra cadeira sapato paralelepipedo macaco'.split()
48 palavra = getPalavra(palavras)
49
50 while fim == False:
51     mostra(erradas, certas, palavra)
52
53     # Pega a letra escolhida pelo jogador
54     chute = getChute(erradas + certas)
55
```



MÃO NA MASSA

- Agora, vamos verificar se a letra digitada pelo usuário está na palavra.
 - Para isso, usamos um simples **if in**.
 - SE o chute estiver certo, então temos que verificar se o jogador já ganhou o jogo.

```
if chute in palavra:
    certas.append(chute)

# Verifica se o jogador ganhou
achouTodas = True
for i in range(len(palavra)):
    if palavra[i] not in certas:
        achouTodas = False
        break
if achouTodas:
    print('Sim! A palavra era "' + palavra + '"! Voce ganhou!')
    fim = True
```



MÃO NA MASSA

- Observe que fazemos um pequeno truque:
 - Supomos que o usuário achou todas, percorremos a palavra letra a letra e, se alguma delas NÃO está na lista de “certas”, mudamos a nossa suposição para Falso e paramos o loop.
- Essa parte foi o caso de a letra ESTAR certa. Mas e se não estiver?
 - Simples, adicionamos um **else**, adicionamos a letra na lista de “erradas” e verificamos se o usuário já usou todas as suas chances.

```
else:  
    erradas.append(chute)  
  
    # Verifica se o jogador ainda tem chances  
    if len(erradas) == chances:  
        mostra(erradas, certas, palavra)  
        print "Voce nao tem mais chances. A palavra correta era: ", palavra  
        fim = True
```



MÃO NA MASSA

- Pronto!! Temos um jogo da Forca escrito em Python!! 😊
- Quando executamos o código fica assim:

```
F O R C A
Letras erradas:
_ _ _ _ _
Chute uma letra.
"a"
Letras erradas:
_ a _ a _ _
Chute uma letra.
"e"
Letras erradas: e
_ a _ a _ _
```

```
Letras erradas: e n
s a p a _ _
Chute uma letra.
"o"
Letras erradas: e n
s a p a _ o
Chute uma letra.
"t"
Sim! A palavra era "sapato"! Uoce ganhou!
```



MÃO NA MASSA

- O código ficou assim:

```
1 import random
2
3 def getPalavra(listaDePalavras):
4     # Retorna uma palavra aleatoria da lista
5     index = random.randint(0, len(listaDePalavras) - 1)
6     return listaDePalavras[index]
7
8 def mostra(erradas, certas, palavra):
9     print 'Letras erradas:',
10    for letra in erradas:
11        print letra,
12    print " "
13
14    espacos = '_' * len(palavra)
15
16    for i in range(len(palavra)): # substitui espacos com as letras corretas
17        if palavra[i] in certas:
18            espacos = espacos[:i] + palavra[i] + espacos[i+1:]
19
20    for letra in espacos: # mostra os espacos com as letras corretas
21        print letra,
22    print " "
23
24 def getChute(usadas):
25     # Retorna a letra digitada e faz a validacao se eh uma letra valida
26     while True:
27         print('Chute uma letra.')
28         chute = input()
29         chute = chute.lower()
30         if len(chute) != 1:
31             print('Por favor informe uma unica palavra')
32         elif chute in usadas:
33             print('Voce ja usou essa letra. Tente novamente')
34         elif chute not in 'abcdefghijklmnopqrstuvwxyz':
35             print('Por favor use apenas LETRAS.')
36         else:
37             return chute
```



```

40 print " F O R C A "
41
42 erradas = []
43 certas = []
44 fim = False
45 chances = 7
46
47 palavras = 'batata barriga bocejo camelo cachorro candelabro cobra cadeira sapato paralelepipedo macaco'.split()
48 palavra = getPalavra(palavras)
49
50 while fim == False:
51     mostra(erradas, certas, palavra)
52
53     # Pega a letra escolhida pelo jogador
54     chute = getChute(erradas + certas)
55
56     if chute in palavra:
57         certas += chute
58
59         # Verifica se o jogador ganhou
60         achouTodas = True
61         for i in range(len(palavra)):
62             if palavra[i] not in certas:
63                 achouTodas = False
64                 break
65         if achouTodas:
66             print('Sim! A palavra era "' + palavra + '"! Voce ganhou!')
67             fim = True
68     else:
69         erradas += chute
70
71         # Verifica se o jogador ainda tem chances
72         if len(erradas) == chances:
73             mostra(erradas, certas, palavra)
74             print "Voce nao tem mais chances. A palavra correta era: ", palavra
75             fim = True

```



OBRIGADA

- Se você chegou até aqui, PARABÉNS! E OBRIGADA pela paciência.
- O código completo do nosso tutorial está em https://github.com/taisbellini/projeto_tutoras
- A documentação em português do Python você pode encontrar em: <http://wiki.python.org.br/>
- Esse site aqui é em inglês e tem tutoriais bem legais usando Python 3 (inclusive o do Jogo da Forca): <http://inventwithpython.com/chapters/>
- Tem várias bibliotecas, extensões e frameworks para Python que facilitam MUITO a nossa vida na hora de criar coisas incríveis, por isso não deixe de pesquisar por extensões para a linguagem.
- Algumas sugestões em inglês: <http://pythontips.com/2013/07/30/20-python-libraries-you-cant-live-without/>



CONTATO

- Se você teve dúvidas durante o tutorial ou gostaria de mais dicas, pode me mandar um email: tais38@gmail.com.
- Sugestões, opiniões, reclamações, ou comentários são muito bem vindos 😊

